

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

In re Application of:)	
)	
Darlet)	
)	
Serial No.: 09/754,785)	Group Art Unit: 2192
)	
Filed: January 4, 2001)	Examiner: Eric B. Kiss
)	
)	Board of Patent Appeals and
For: SYSTEM AND METHOD FOR)	Interferences
LINEAR PROCESSING OF)	
SOFTWARE MODULES)	
)	
Confirmation No.: 3238)	

Mail Stop: Appeal Brief – Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

REPLY BRIEF UNDER 37 C.F.R. § 41.41

In response to the Examiner's Answer mailed on September 15, 2009, to the Appeal Brief filed on July 9, 2009, and pursuant to 37 C.F.R. § 41.41, Appellant presents this reply brief in the above-captioned application.

This is an appeal to the Board of Patent Appeals and Interferences from the Examiner's final rejection of claims 1-15 and 40-60 in the Final Office Action dated December 31, 2008. The appealed claims are set forth in the attached Claims Appendix.

1. Status of the Claims

Claims 1-15 and 40-60 have been rejected in the Final Office Action. Claims 16-39 were canceled in a previous amendment. The final rejection of claims 1-15 and 40-60 is being appealed.

2. Ground of Rejection to be Reviewed on Appeal

- I. Whether claims 1-15, 40, 41, and 43-60 are unpatentable over 35 U.S.C. § 103(a) by “Linkers and Loaders, chapter 6” by John Levine, June 1999 (hereinafter “Levine”).
- II. Whether claim 42 is unpatentable over 35 U.S.C. § 103(a) by Levine in view of U.S. Patent No. 6,185,733 to Breslau et al. (hereinafter “Breslau”).

3. Argument

- I. Claims 1-15, 40, 41, and 43-60 should not be rejected under 35 U.S.C. § 103(a) based on “Linkers and Loaders, chapter 6” by Levine.

Claim 1 recites “[a] computer readable storage medium including a set of instructions executable by a processor, the set of instructions operable to: receive a software module, the software module including references to locations within the software module, at least some of the references being backward references; and reorder components of the software module into a predetermined order based on a type of the components to remove at least some of the backward references, wherein the components include at least one of a header, a section, and a table, wherein the reordered software module includes the at least some of the backward references, and wherein the at least some of the backward references in the reordered software module are stored in a memory to avoid a nonsequential reading of the reordered software module.”

In the Appeal Brief, Appellant argued that *Levine* does not teach reordering components of a software module “into a *predetermined order* based on a *type* of the components.” The Examiner argues the following:

The purpose of the topological sort disclosed by *Levine* is to sort the input files such that there are no backward references (symbols defined before references to them), and this sorted list is processed by *ar* to store the recorded object code... The output of the *lorder* command is a “predetermined order”, i.e., an ordered list of files with no backwards references, and this predetermined order is issued to the *ar* command to produce the actual reordered output.

Examiner’s Answer at page 10. In this argument, by using the “i.e.” to equate “predetermined order” with “an ordered list of files...,” the Examiner elides over the prefix “pre.” It is not the case that simply ordering a list of files means that the ordering has been done according to a predetermination of that order. This is where the Examiner has not established that the evidence, as represented by *Levine*, addresses this issue. The specification, as explained in the Appeal Brief, lays out a predetermined ordering of the components. For example, in the exemplary embodiment of Figures 1-4 of the Specification of the present invention, the predetermined order recited includes a module header 310, a program header table 320, a section header table 330, a section string table 340, an entry point table 350, a text section 360, one or more data sections 370, a symbol string table 380, a symbol table 390, and a relocation information table 395. (See Specification, pp. 5-17, Figs. 1-4.) Thus, prior to the creation of an actual order of components, the claimed invention has established a template of sorts to which the ordering process is to adhere. This is in contrast to what happens in *Levine*, which instead of arranging files according to a predetermined ordering, does so dynamically determined based on symbol references contained within the various components. (See *Levine*, pp. 5-6, § “Creating libraries”). So, by failing to describe a process of arranging the input files according to which, before any ordering of the components occurs, a predetermination is made regarding the particular ordering of the components, *Levine* cannot be regarded as involving any predetermination. If the Examiner believes that such a feature is somehow inherent to the ordering in this reference, the burden is on the Examiner to demonstrate that the ordering in *Levine* is necessarily based on a predetermination. Unless such a demonstration is made, it is not the case that the Examiner has

adequately proven that this reference involves any kind of predetermination in terms of the ordering of its files.

The next issue is whether it is the case that *Levine* shows a reordering based on the type of the components. The Examiner finds Appellant's argument unpersuasive because, according to the Examiner, Appellant has argued that ordering according to a type means ordering according to the function of the various components, which, in the Examiner's opinion, impermissibly reads into the claims subject matter to which the claim is not necessarily limited. The argument of the Examiner is that the specification contains no definition of "type" that is set forth with such clarity, deliberateness, and precision as to require it to be interpreted to mean the function of the components. The Examiner then points to various portions of the specification demonstrating, according to the Examiner, that the applicant has not adopted for the word "type" a single definition that would warrant its being incorporated into the meaning of the claim term.

The plain language of the claim means that the reordering is accomplished according to the type of the component. In *Levine*, an ordering of object files is accomplished, but without reference to the type of object files being reordered. In fact, the Examiner has not established that there is any kind of object file other than a single type. According to *Levine*, "a library file is fundamentally no more than a collection of object files." *Id.* at page 1. Nevertheless, *Levine* does not provide any support for the notion that these object files cover multiple types of such files. At the very least, it is not the case that the Examiner has established that anything more than a single, generic type of object file is used in *Levine*. If that is the case, namely, that only one type of object file is taught by this reference, then it follows that a reordering according to type cannot be accomplished, since it makes no sense to reorder according to a single type.

4. Conclusion


For the reasons set forth above, Appellant respectfully requests that the Board reverse the above-discussed rejections of claims 1-15 and 40-60.

Respectfully submitted,

Date:

11/16/09

By:


Michael J. Marcin (Reg. No. 48,198)

Fay Kaplun & Marcin, LLP
150 Broadway, Suite 702
New York, NY 10038
Tel.: (212) 619-6000
Fax: (212) 619-0276

CLAIMS APPENDIX

1. (Previously Presented) A computer readable storage medium including a set of instructions executable by a processor, the set of instructions operable to:
 - receive a software module, the software module including references to locations within the software module, at least some of the references being backward references; and
 - reorder components of the software module into a predetermined order based on a type of the components to remove at least some of the backward references,
 - wherein the components include at least one of a header, a section, and a table,
 - wherein the reordered software module includes the at least some of the backward references, and
 - wherein the at least some of the backward references in the reordered software module are stored in a memory to avoid a nonsequential reading of the reordered software module.
2. (Previously Presented) The computer readable storage medium according to claim 1, wherein the set of instructions is further operable to:
 - adjust at least one of the references in the software module to reflect the reordering of the components of the software module, so that the at least one of the references remains a reference to the same component, but to the component's new, reordered location, the new, reordered location coming after the at least one reference in the software module.
3. (Previously Presented) The computer readable storage medium according to claim 2, wherein the software module includes a symbol table, the symbol table including backward references when the reordering of the components of the software module and adjusting the at least one of the references have been completed.
4. (Previously Presented) The computer readable storage medium according to claim 2, wherein the software module includes a symbol table, the software module including no backward references in locations before the symbol table when the reordering of the components of the software module and adjusting the at least one of the references have been completed.

5. (Previously Presented) The computer readable storage medium according to claim 2, wherein the software module is a relocatable object code module in ELF format when the reordering the components of the software module and adjusting the at least one of the references have been completed.

6. (Previously Presented) The computer readable storage medium according to claim 5, wherein, when the software module is received, the software module is a relocatable object code module in ELF format, and wherein, when the reordering the components of the software module and adjusting the at least one of the references have been completed, the software module includes a symbol table, the symbol table including backward references, and the software module includes no backward references from locations before the symbol table.

7. (Previously Presented) The computer readable storage medium according to claim 1, wherein the software module comprises at least one segment, each at least one segment comprising at least one section, and wherein sections in the same segment are contiguously located in the software module when the reordering of the components of the software module has been completed.

8. (Previously Presented) The computer readable storage medium according to claim 1, wherein, when the software module is received, the software module is a relocatable object code module in ELF format.

9. (Previously Presented) A system, comprising:

a memory storing a reorder module configured to receive a software module including references to locations within the software module, at least some of the references being backward references, the reorder module configured to reorder components of the software module into a predetermined order based on a type of the components and remove at least some of the backward references, the components including at least one of at least one of a header, a section, and a table; and

a processor executing the reorder module, wherein the reordered software module includes the at least some of the backward references, and

wherein the at least some of the backward references in the reordered software module are stored in a memory to avoid a nonsequential reading of the reordered software module.

10. (Previously Presented) The system according to claim 9, wherein the reorder module is configured to adjust a reference in the software module to reflect the reordering of the components of the module.

11. (Original) The system according to claim 9, wherein the software module includes a symbol table, and wherein the reorder module is configured not to remove backward references from the symbol table.

12. (Original) The system according to claim 9, wherein the software module includes a symbol table, and wherein the reorder module is configured to remove all backward references from locations before the symbol table in the reordered software module.

13. (Original) The system according to claim 9, wherein the software module includes at least one segment, each of the at least one segments including at least one section, and the reorder module is configured to locate sections in the same segment contiguously in the reordered software module.

14. (Original) The system according to claim 9, wherein the software module is a relocatable object code module in ELF format, and the reordered software module is a relocatable object code module in ELF format.

15. (Previously Presented) The system according to claim 14, wherein the software module includes a symbol table, wherein the reorder module is configured to adjust a reference in the software module to reflect the reordering of the components of the module, wherein the reorder module is configured to remove all backward references from locations before the symbol table, and wherein the reorder module is configured not to remove backward references from the symbol table.

16. (Cancelled) .

17. (Cancelled).

18. (Cancelled).

19. (Cancelled).

20. (Cancelled).

21. (Cancelled).

22. (Cancelled)

23. (Cancelled).

24. (Cancelled).

25. (Cancelled).

26. (Cancelled).

27. (Cancelled).

28. (Cancelled).

29. (Cancelled).

30. (Cancelled).

31. (Cancelled).

32. (Cancelled).

33. (Cancelled).

34. (Cancelled).

35. (Cancelled).

36. (Cancelled).

37. (Cancelled).

38. (Cancelled).

39. (Cancelled).

40. (Previously Presented) The computer readable storage medium of claim 1, wherein the reordering of the components of the software module is completed prior to linking the software module.

41. (Previously Presented) The computer readable storage medium of claim 40, wherein the set of instructions is further operable to:
link the reordered software module.

42. (Previously Presented) The computer readable storage medium of claim 1, wherein the set of instructions is further operable to:
transfer the reordered software module to a different computer system; and linking the reordered software module on the different computer system.

43. (Previously Presented) The computer readable storage medium of claim 1, wherein the reordered components include an ELF data section.
44. (Previously Presented) The computer readable storage medium of claim 1, wherein the reordered components include an ELF code section.
45. (Previously Presented) The computer readable storage medium of claim 1, wherein the reordered components include an ELF header table.
46. (Previously Presented) The computer readable storage medium of claim 1, wherein the reordered components include an ELF entry point table.
47. (Previously Presented) The computer readable storage medium of claim 1, wherein the reference points to a section located prior to the reference in the received software module.
48. (Previously Presented) The computer readable storage medium of claim 47, wherein, after the software module has been reordered, the reference is changed to point at the same section, the section having been relocated to appear after the reference in the reordered software module.
49. (Previously Presented) The computer readable storage medium of claim 1, wherein the reference points to a table located prior to the reference in the received software module.
50. (Previously Presented) The computer readable storage medium of claim 49, wherein, after the software module has been reordered, the reference is changed to point at the same table, the table having been relocated to appear after the reference in the reordered software module.
51. (Previously Presented) The computer readable storage medium of claim 1, wherein the reference points into a section located prior to the reference in the received software module.

52. (Previously Presented) The computer readable storage medium of claim 51, wherein, after the software module has been reordered, the reference points into the same section, the section having been relocated to appear after the reference in the reordered software module.
53. (Previously Presented) The computer readable storage medium of claim 1, wherein the reference points into a table located prior to the reference in the received software module.
54. (Previously Presented) The computer readable storage medium of claim 53, wherein, after the software module has been reordered, the reference is changed to point into the same table, the table having been relocated to appear after the reference in the reordered software module.
55. (Previously Presented) A computer readable storage medium including a set of instructions executable by a processor, the set of instructions operable to:
- receive a software module, the software module including components arranged in a first order, a first one of the components including a reference to a location in a second one of the components, the second one of the components preceding the first one of the components in the first order; and
 - arrange the components into a predetermined second order to produce a reordered software module so that the second one of the components is subsequent to the first one of the components in the second order, wherein the arrangement is based on a type of the first and second ones of the components,
 - wherein the components include at least one of a header, a section, and a table,
 - wherein the reordered software module includes at least one reference from a third component to a preceding component, and
 - wherein the at least one reference from the third component is stored in a memory to avoid a nonsequential reading of the reordered software module.
56. (Previously Presented) The computer readable storage medium of claim 55, wherein the arranging occurs prior to linking the software module.

57. (Previously Presented) The computer readable storage medium of claim 56, wherein the set of instructions is further operable to:

linking the software module without storing the entire software module in local memory.

58. (Previously Presented) The computer readable storage medium of claim 57, wherein the components include an ELF table and an ELF section.

59. (Previously Presented) The computer readable storage medium of claim 58, wherein the order of segments within the ELF section is preserved when the section is moved to a different position in the reordered software module.

60. (Previously Presented) The computer readable storage medium of claim 59, wherein the only backward references between different ELF components in the reordered software module are references located in the ELF symbol table.

EVIDENCE APPENDIX

No evidence has been submitted herewith or is relied upon in the present appeal.

RELATED PROCEEDINGS APPENDIX

There are no related proceedings or decisions which relate to the present appeal.